Linus Gasser
Lead Engineer
Center for Digital Trust (C4DT)

ergon

Roman Wixinger
Data Scientist
Data & AI Team

Re-runnable code is all you need

How to create reproducible and shareable computing environments

RSECon2024

EPFL
ergon

Re-Runnable Code is All You Need

1. **Introduction**
2. Deep Dive Devbox
3. Case Studies
4. Conclusion

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

Introduction

Which terminology do we need in this talk?

Why does re-runnable code matter to us us?

How to select the right tooling?

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

Introduction: Terminology

According to [1], research code should …

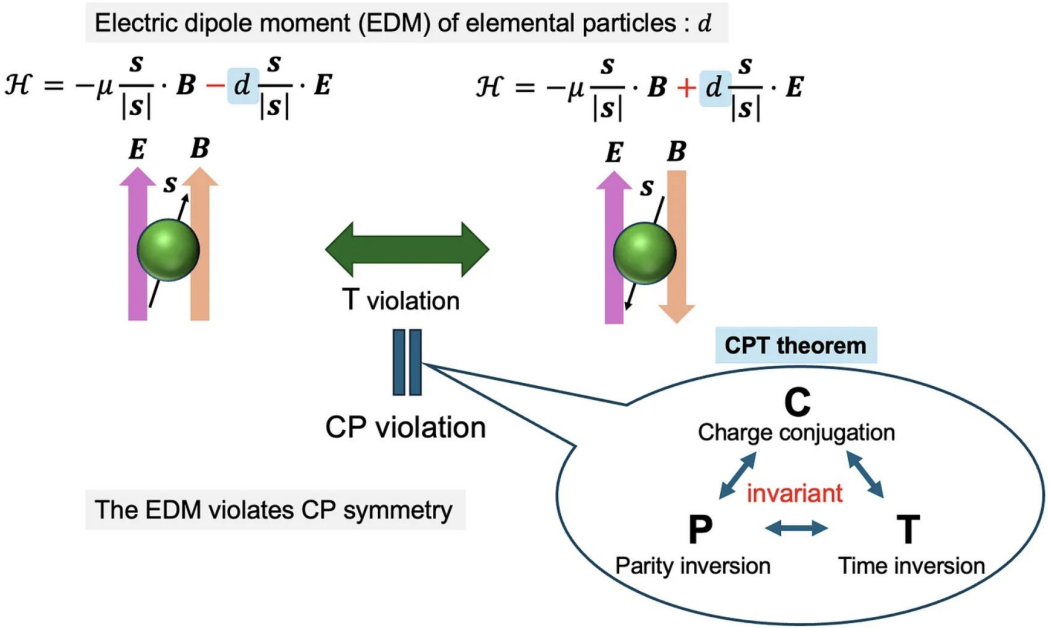
1. **Re-runnable**: be executable.
2. **Repeatable**: produce the same result more than once.
3. **Reproducible**: allow an investigator to reobtain the published results.
4. **Reuseable**: be easy to use, understand and modify.
5. **Replicable**: act as as an available reference for any ambiguity in the algorithmic descriptions of the article.

[1] Benureau, F. C., & Rougier, N. P. (2018). Re-run, repeat, reproduce, reuse, replicate: transforming code into scientific contributions. Frontiers in neuroinformatics, 11, 69.

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch


三の丸尚蔵館 大手門

Experiment: Matter-antimatter asymmetry in the universe



In case an Electric Dipole Moment (EDM) of elemental particles is found, then it violates the T symmetry (time) of the system. According to the CPT theorem, this would imply that the CP symmetry (charge and parity) is violated too. CP symmetry breaking is one of the candidate theories for explaining why there is a matter-antimatter asymmetry in the universe [2].

[2] A. D. Sakharov (1967). Violation of CP Invariance, C Asymmetry and Baryon Asymmetry of Universe, Journal of Experiment and Theoretical Physics Letters 5, 24-27.

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch


DT-01 CE 0598

Experiment: What happens when our code is not re-runnable

Symptoms:
- Automation becomes hard
- Collaboration is challenging

Possible root causes:
- Crucial steps are not defined in code
- Code is not designed to be shareable
- Environment variables are not managed well

[3] Blog post https://engineersforscience.substack.com/p/utokyo-case-study

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

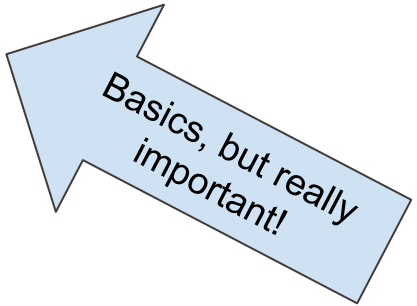
Experiment: The solution is often much simpler than we think ...

Solution:

- Define all steps in code
- Use proper package management
- Manage environment variables with configs

Take away:

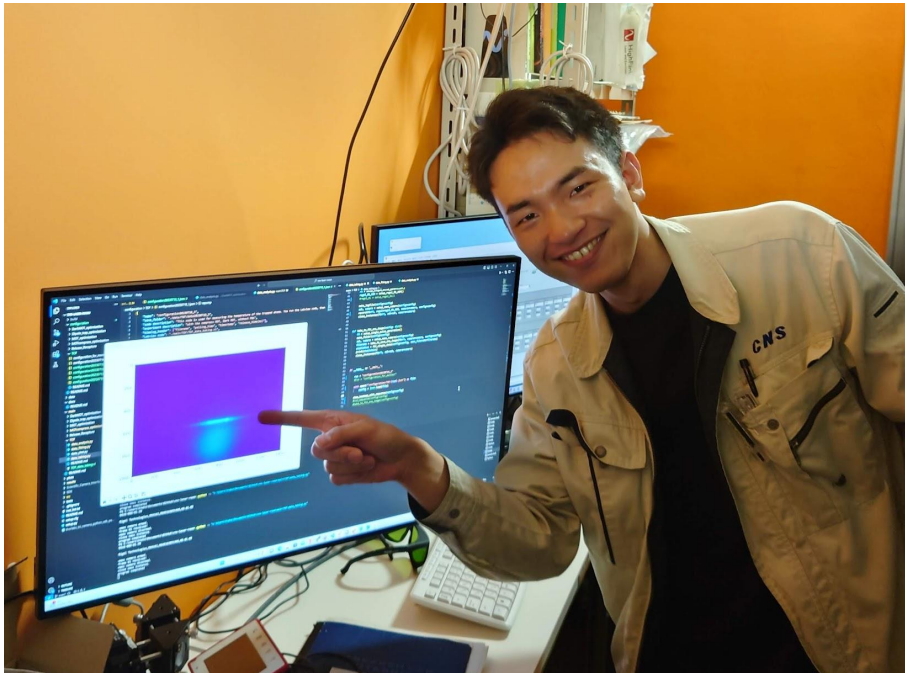
- When we invest in the basics, they will help us everyday
- We use tooling to solve a problem, not because it is cool



[3] Blog post https://engineersforscience.substack.com/p/utokyo-case-study

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

Experiment: … and makes the experiment successful!



July 2023: Nagase-san with the first atoms he captured with his Optical Dipole Trap

[3] Blog post https://engineersforscience.substack.com/p/utokyo-case-study

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Introduction: Terminology

1. **Version control**: Track changes and collaborate on code (Git).
2. **Package management**: Install and manage software libraries (pip).
3. **Environment management**: Ensure consistent software environments (devbox).
4. **Containerization**: Isolate software and dependencies (Docker).
5. **Workflow automation**: Automate repetitive tasks (GitHub Actions).
6. **Configuration management**: Maintain consistent software settings (Ansible).
7. **Infrastructure as Code (IaC)**: Code-based infrastructure setup. (Terraform).

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Introduction: Evaluating tools

Key factors for selecting an environment management tool:

- **Ease of Use**: Simple setup, intuitive commands, and minimal learning curve.
- **Integration with Version Control**: Seamless syncing with tools like Git.
- **Cross-Platform Compatibility**: Consistent behavior across macOS, Windows (WSL), and Linux.
- **Dependency Compatibility**: Supports diverse libraries and tools without conflicts.
- **Offline Usability**: Fully functional without internet access.
- **Performance**: Efficient resource usage; scales with project complexity.
- **Cost Efficiency**: Low or manageable costs for setup and maintenance.

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Re-Runnable Code is All You Need

1. Introduction
2. **Deep Dive Devbox**
3. Case Studies
4. Conclusion

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Deep Dive Devbox

- Re-runnable environments types
- Devbox - what is it?
  - How does it work?
  - Comparison with docker workflow
- Simple use-case
  - Different version of Node.js
  - Commands
- Github actions
- Usage in docker

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

		Binary	
		Same	Different
Problem Domain	Same	**Reproducible**	**Re-runnable**
	Different	**Useful**	**Generalisable**

Copied and adapted from Richard Fitz John's talk on Tuesday

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

What do you use for re-runnable environments?

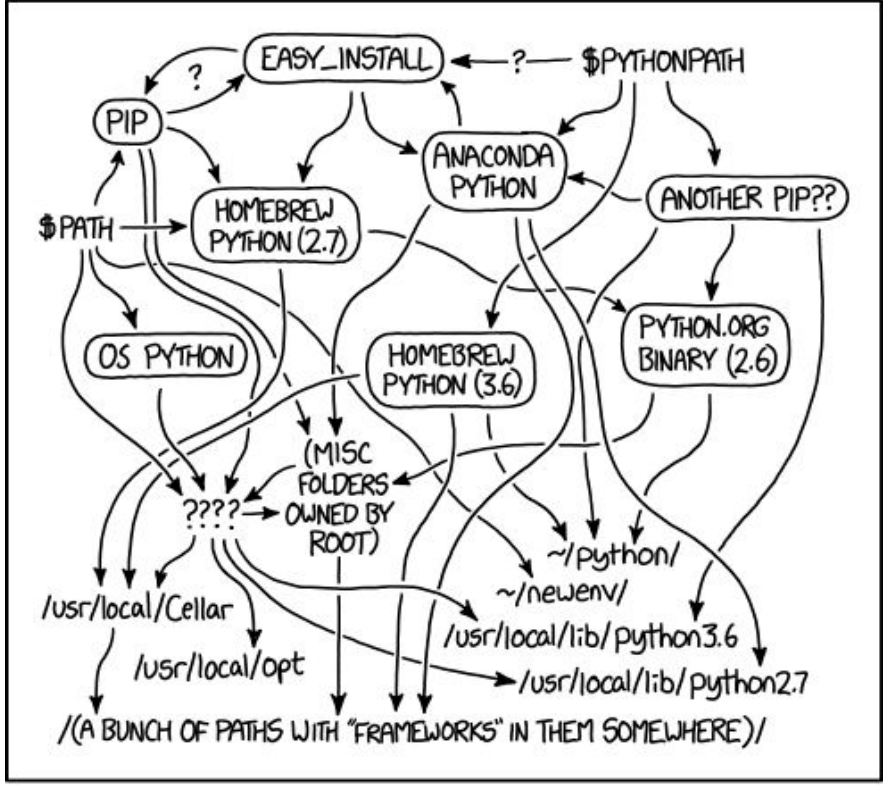
Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

Some Re-runnable Environment Types

Language-specific	Developer Machines (mostly using PATH)	Binaries for Users (Virtualizations, chroot, …)	High Performance Computing
• nvm • gvm • pip / pyenv • maven	• devbox / flox • mise • asdf • brew / chocolatey • apt / yum / apk / rpm	• Docker • Flatpack • Snap	• spack • easybuild

Plus of course your personal favorite which is not on the list :)

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

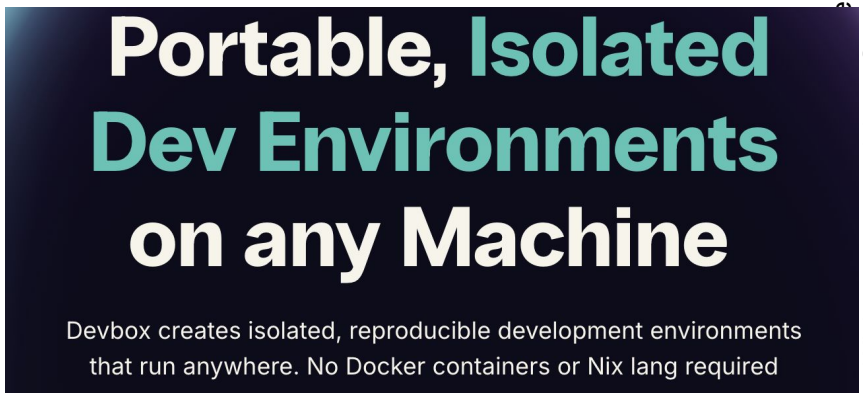


MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

The Python environmental protection agency wants to seal it in a cement chamber, with pictorial messages to future civilizations warning them about the danger of using sudo to install random Python packages. - https://xkcd.com/1987/

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

Devbox - TLDR;



Portable: runs on your laptop, github workflows, your customers' laptop, …

Isolated Dev Environment: use *that* version of node, golang, rust, java, etc. - isolation per shell

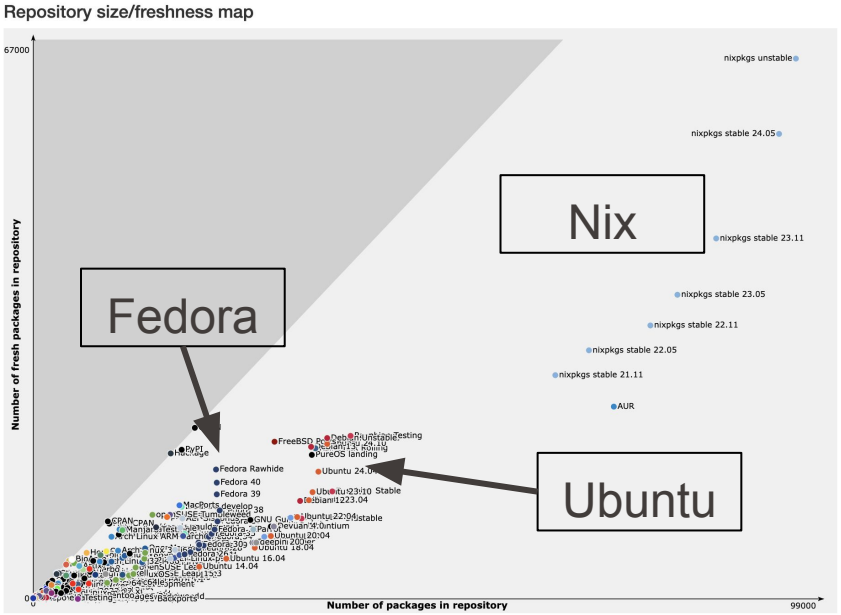
On any Machine: works on Linux, Mac, Windows WSL

Many many packages: over 400,000 versions of 80,000 packages powered by Nix

Roman Wixinger, roman.wixinger@ergon.ch

How does it work?

- Uses packages from Nix
  - Name it, nix got it
  - Installation is per-directory
- Versions are stored in *devbox.json* and *devbox.lock*
- Run your project in the `devbox shell` Environment
- Devbox sets PATH variable to those versions
  - Can be automated using `direnv`



EPFL
ergon

Simple Use-Case 1/2

Project with nodejs version 20, another with version 22

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

How do you separate node versions for development?

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Simple Use-Case 1/2

Project with nodejs version 20, another with version 22

- Create directory and initialize devbox
  ```
  $ mkdir clash20 && cd clash20 && devbox init
  ```
  This creates an example *devbox.json*
- Install some packages
  ```
  $ devbox add nodejs@20 vim git
  ```
  Devbox either uses the local cache at /nix, or downloads the packages.
- Start using this environment
  ```
  $ devbox shell --pure # Don't include system cmds
  $ node --version
  v20.15.1 # At the time of this writing
  ```


Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Simple Use-Case 2/2

Other project with nodejs version 22

- Create directory and initialize devbox

```
$ mkdir clash22 && cd clash22 && devbox init
$ devbox add nodejs@22 vim git
$ devbox shell --pure
$ node --version
v22.5.1
```

- Check nodejs version of other project

```
$ exit
$ cd ../clash20
$ devbox run -- node --version
v20.15.1
```


Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

devbox.json

```
{
  "$schema":
"https://raw.githubusercontent.com/.../devbox.schema.json",
  "packages": [
    "nodejs@22",
    "git@latest",
    "vim@latest"
  ],
  "shell": {
    "init_hook": [
      "npm ci"
    ],
    "scripts": {
      "test": [
        "npm test"
      ] } } }
```

It's a JSON-LD!

Packages and versions

Startup commands

Allows for simple scripting

To run it:

```
devbox run test
```

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Commands / Scripting

Similar to a very simple Makefile

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Did you ever write a Makefile from scratch?

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL ergon

Commands / Scripting

Similar to a very simple Makefile:

- Environment variables for
  - Directory colors for *ls* with dark background
  - Run commands installed with *npm ci*
- Initialization when starting:
  - npm packages *npm ci*
  - make an alias for *ls*
- Three commands:
  - test - runs the tests with *npm test*
  - serve - serves the project with *ng serve*
  - build - builds the project with *ng build*

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

devbox.json

```
"env": {
  "LS_COLORS": "di=33",
  "PATH": "$PATH:$PWD/node_modules/.bin"
},
"shell": {
  "init_hook": [
    "alias ls='ls --color'",
    "npm ci"
  ],
```

```
"scripts": {
  "test": [
    "npm test"
  ],
  "serve": [
    "ng serve"
  ],
  "build": [
    "ng build"
  ],
}
```


Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Running commands

Now you can do a

`devbox run (test|serve|build)`

on any machine with devbox installed!

Or use

`devbox shell [--pure]`

to continue development.

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Github Actions

```yaml
name: Testing with devbox

on: push

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Install devbox
        uses: jetify-com/devbox-install-action@v0.9.0

      - name: Run tests
        run: devbox run test
```


Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

And more

Create Dockerfile:

- `devbox build dockerfile`

Manage services:

- similar to docker-compose.yaml

Some paying options:

- Run in the cloud
  - devbox.sh/github.com/ineiti/fledger
- Run devcontainers

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Re-Runnable Code is All You Need

1. Introduction
2. Deep Dive Devbox
3. **Case Studies**
4. Conclusion

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

Do you use a full CI/CD pipeline?
(Testing - Package publishing - Deploying)

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

Case study - github.com/ineiti/fledger

Using devbox to:

- Easier onboarding of students
  - Install the rust toolchain
  - Include the wasm rust toolchain
- Only one source of tool versions
  - local installation vs. github-actions
- Github actions for CI/CD
  - Testing
  - Building docker files
    - Compiling in github action to use github cache for rust
  - Updating live servers (unchanged)

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

Fledger - devbox.json rustup installation

```
{
  "packages": { "rustup": "latest", "trunk": "latest",
"wasm-pack": "latest" },
  "shell": {
    "init_hook": [
      "if [ ! -d $RUSTUP_HOME/toolchains/stable* ]; then
rustup default stable; fi",
      "if [ ! -d
$RUSTUP_HOME/toolchains/stable*/lib/rustlib/wasm32-unknown-u
nknown ]; then rustup target add wasm32-unknown-unknown;
fi",
    ],
    }
  }
```


Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

Fledger - single source of versions

Same version **locally**, in **github actions**, and in **Dockerfiles**:

```
-     - name: Install trunk
-       run: which trunk || cargo install --locked trunk
-
-     - uses: jetli/wasm-pack-action@v0.3.0
+     - name: Install devbox
+       uses: jetify-com/devbox-install-action@v0.11.0
        with:
```


Roman Wixinger, roman.wixinger@ergon.ch

EPFL ergon

Fledger - github workflow

Replace direct calls with

`devbox run --`

so they use devbox

Profit!

```
      - name: Run cargo_build
-       shell: bash
-       run: |
-         make cargo_build
+       run: devbox run -- make cargo_build
```

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Fledger - Rust Linking Devbox Glibc

Problem:

```
$ file flsignal
...ELF 64-bit...interpreter
/nix/store/kz7xglxzhad64v667wwpn8vrxhjwcbna-glibc-2.39-52
/lib/ld-linux-x86-64.so.1
```

So it doesn't run in a minimal docker image.

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Fledger - Building Dockerfiles

```
FROM debian:bookworm-slim

RUN apt update && apt install patchelf

COPY target-common/release/flsignal flsignal
RUN patchelf --set-interpreter /usr/lib64/ld-linux-x86-64.so.2 flsignal

FROM debian:bookworm-slim
WORKDIR /fledger
COPY --from=0 flsignal /fledger/flsignal

ENTRYPOINT ["/fledger/flsignal", "-vv"]
```


Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Fledger - rust/devbox shenanigans

```
{
  "packages": {...
    "darwin.apple_sdk.frameworks.SystemConfiguration": {
      "platforms": ["x86_64-darwin", "aarch64-darwin"]
    },
    "darwin.apple_sdk.frameworks.Security": {
      "platforms": ["x86_64-darwin", "aarch64-darwin"]
    },
  },
```


Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL ergon

Case Study - Tutorial Java Project [4]

- Easy installation
- Maven packages installation
  - per project - more space, but separation
  - global - faster installation, might collide

Some problems:

- How to run multi-line github actions with devbox?
  - Use devbox scripts

[4] Tutorial by Jaime Cardozo from the ETH Library

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

Most difficult language re-runnable wise?

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

The good, the bad, and the ugly

The Good

- Very simple configuration file
- Based on nix - huge collection of packages
- direnv can automate it even further

The Bad

- JSON configuration files - YAML would be nicer (comments, multi-line scripts)
- Some packages differ between platforms (e.g., psutils)
- Non-system compiler (openssl, rust-Security)
- No common shell history

The Ugly

- Non-root installation of devbox and nix is buggy

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL
ergon

Re-Runnable Code is All You Need

1. Introduction
2. Deep Dive Devbox
3. Case Studies
4. **Conclusion**

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL ergon

Benefits of proper environment management

- **Time savings**: Streamlines onboarding, allowing new team members to get started faster.
  - Tip: A thorough onboarding document can save weeks of time.
- **Scientific impact**: Enhances the adoption of research findings and code by others.
  - Insight: This is based on our experience.
- **Productivity**: Reduces errors from version mismatches and environment issues.
- **Portability**: Makes research code more portable, enabling automation and cross-domain applications.

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

Conclusion

- Environment management isn’t glamorous, but the productivity gains are!
- Devbox simplifies managing multiple environments by isolating them on a per-folder basis.
- For deployment, Devbox integrates seamlessly with Docker, eliminating the need to list endless dependencies in the Dockerfile.
- As RSEs, we have a responsibility to empower students and researchers with the right tools for effective and reproducible research.

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

Linus Gasser
Lead Engineer
Center for Digital Trust (C4DT)

Roman Wixinger
Data Scientist
Data & AI Team

Re-runnable code is all you need

Special thanks go to Jaime Cardozo!

Linus Gasser
Lead Engineer
Center for Digital Trust (C4DT)

Roman Wixinger
Data Scientist
Data & AI Team

Re-runnable code is all you need

Thank you!

RSECon2024

EPFL
ergon

Backup slides

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

EPFL ergon

Devbox vs. Dockerfile

Dockerfile

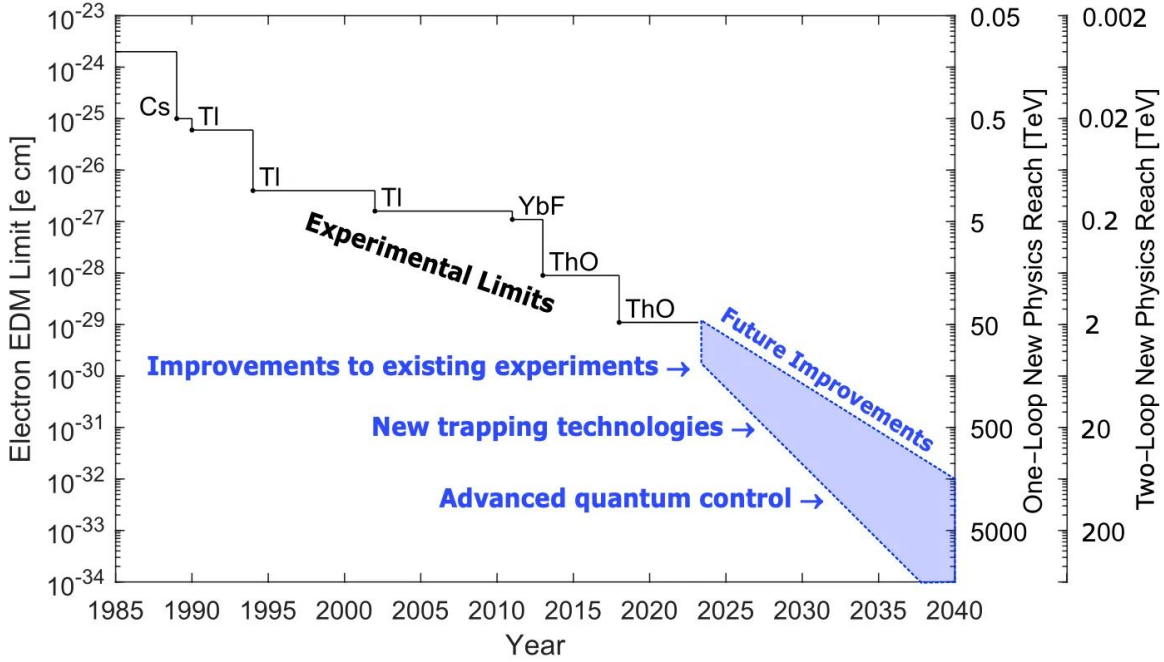
- Uses linux namespaces, with a linux VM on Windows / Mac
- *FROM* base package
  - Use package manager
  - Add packages
- Produces a docker image with all binaries
- Ready-to use application

Devbox

- Uses the system shell and packages compiled for the system
- Installs packages in sub-directories of */nix*
- Produces *devbox.json* and *devbox.lock* with references to installed packages
- Ready-to use development environment

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

Experiment: Testing the Standard Model with cold atoms vs. colliders



Upper bound on the electron EDM over time, shown with the corresponding energy scale needed to observe new physics based on one-loop and two-loop effects. For comparison, colliders at CERN currently achieve a maximum collision energy of 13 TeV [2].

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

[2] Chupp, T. E., Fierlinger, P., Ramsey-Musolf, M. J., & Singh, J. T. (2019). Electric dipole moments of atoms, molecules, nuclei, and particles. Reviews of Modern Physics, 91(1), 015001.

Case Studies

Real-life examples

- github.com/c4dt/rse.epfl.ch
  - Used by other members of EPFL
  - Hugo and rclone (plus some others)
  - Configuration for log-in is local and not in github…
- github.com/ineiti/fledger
  - Student project based on rust
  - Complete rust environment, including wasm
  - Docker-image generation

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

rse.epfl.ch - goals

Our local RSE page:

- Using hugo
  - need the correct version
  - install the theme (git submodule)
  - let the user call hugo
- Pushing using rclone
  - Configuration of passwords is local
- Don't forget the README.md!

Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

rse.epfl.ch - hugo themes

```
{
  "packages": [... ],
  "shell": {
    "init_hook": [
      "if [ ! -f themes/anake/theme.toml ]; then git
submodule init && git submodule update; fi",
      "alias ls='ls --color'",
      "alias ll='ls -l --color'"
    ],
```


Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

rse.epfl.ch - command line arguments

```
{
  "packages": [... ],
  "shell": {
    "init_hook": [
      "if [ ! -f
themes/anake/theme.toml ]; then
git submodule init && git
submodule update; fi",
      "alias ls='ls --color'",
      "alias ll='ls -l
--color'"
    ],
```

```
"scripts": {
  "hugo": [
    "hugo $@"
  ],
  "server": [
    "hugo server -D"
  ],
  "update": [
    "git pull",
    "hugo",
    "rclone copy public/
ic-ftps:/rse.epfl.ch/"
  ]
```


Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch

rse.epfl.ch - multi-line script

```
{
  "packages": [... ],
  "shell": {
    "init_hook": [
      "if [ ! -f
themes/anake/theme.toml ]; then
git submodule init && git
submodule update; fi",
      "alias ls='ls --color'",
      "alias ll='ls -l
--color'"
    ],
```

```
"scripts": {
  "hugo": [
    "hugo $@"
  ],
  "server": [
    "hugo server -D"
  ],
  "update": [
    "git pull",
    "hugo",
    "rclone copy public/
ic-ftps:/rse.epfl.ch/"
  ]
```


Linus Gasser, linus.gasser@epfl.ch
Roman Wixinger, roman.wixinger@ergon.ch